Investigating Stochastic Diffusion Search in Data Clustering

Mohammad Majid al-Rifaie*¶‖, Daniel Joyce†‖, Sukhi Shergill‡‖, Mark Bishop§¶
¶Department of Computing, Goldsmiths, University of London, London SE14 6NW, United Kingdom
‖Cognition, Schizophrenia & Imaging Laboratory, King's College London, London SE5 8AF, United Kingdom
*m.majid@gold.ac.uk / m.majid@kcl.ac.uk
†daniel.joyce@kcl.ac.uk
†sukhi.shergill@kcl.ac.uk
§m.bishop@gold.ac.uk

***Abstract*—The use of clustering in various applications is key to its popularity in data analysis and data mining. Algorithms used for optimisation can be extended to perform clustering on a dataset. In this paper, a swarm intelligence technique – Stochastic Diffusion Search – is deployed for clustering purposes. This algorithm has been used in the past as a multi-agent global search and optimisation technique. In the context of this paper, the algorithm is applied to a clustering problem, tested on the classical Iris dataset and its performance is contrasted against nine other clustering techniques. The outcome of the comparison highlights the promising and competitive performance of the proposed method in terms of the quality of the solutions and its robustness in classification. This paper serves as a proof of principle of the novel applicability of this algorithm in the field of data clustering.**

***Keywords*—Clustering; Stochastic Diffusion Search; iris dataset; Swarm intelligence**

I. Introduction

Given the undeniable significance of data clustering in different and diverse scientific domains (e.g. computer science, psychology, medicine), various techniques have been proposed over the years. Nature-inspired metaheuristic algorithms are among one of the categories which aimed at providing solutions to this problem. In this paper a novel method in addressing data clustering problems is used where a swarm intelligence algorithm is adapted for this purpose.

This work attempts to pave the way for more effectively optimising computationally expensive objective functions, by deploying the diffusion mechanism of Stochastic Diffusion Search [1], [2] to more efficiently allocate resources via information-sharing between agents. A recent review [2] details the extensive applications of this algorithm in the last two decades in various fields (e.g. optimisation, resource allocation, medical imagining, etc).

This paper focuses on using Stochastic Diffusion Search (SDS) in data clustering in order to provide a proof of principle. In this paper, first Stochastic Diffusion Search algorithm is explained, followed by a simple example detailing its behaviour and highlighting one of its main features (i.e. partial function evaluation). Afterwards, its application into a well-known clustering problem is reported, and finally the results are compared against other methods used in the area.

II. Stochastic Diffusion Search

Stochastic Diffusion Search (SDS) [2], [1] which was first proposed in 1989 is a probabilistic approach for solving best-fit pattern recognition and matching problems. SDS, as a multi-agent population-based global search and optimisation algorithm, is a distributed mode of computation utilising interaction between simple agents. Its computational roots stem from Geoff Hinton's interest 3D object classification and mapping.

The SDS algorithm commences a search or optimisation by initialising its population and then iterating through two phases: the test and diffusion phases.

In the test phase, SDS checks whether the agent hypothesis is successful or not by performing a hypothesis evaluation which returns a boolean value[1]. Once the activity (i.e their status as being 'true' or 'false') of all the agents are determined, successful hypotheses diffuse across the population and in this way information on potentially good solutions spreads throughout the entire population of agents. In other words, each agent recruits another agent for interaction and potential communication of hypothesis. The spreading of information occurs during the diffusion phase. See Algorithm 1 for a high-level description of the algorithm.

Algorithm 1 SDS Algorithm

```
01: Initialise agents
02: While (stopping condition is not met)
04:    For each agent
03:        Test hypothesis and determine activity
05:    For each agent
06:        Diffuse hypothesis
07: End While
```

A. Standard SDS and Passive Recruitment

In standard SDS (which is used in this paper), *passive recruitment mode* is employed. In this mode, if the agent is inactive, a second agent is randomly selected for diffusion; if the second agent is active, its hypothesis is communicated (*diffused*) to the inactive one. Otherwise there is no flow of information between agents; instead a completely new hypothesis is generated for the first inactive agent at

[1]A hypothesis can be considered a point in the search space. Later in the paper hypothesis is explained in the context of the clustering problem

random (see Algorithm 2). Therefore, recruitment is not the responsibility of the active agents. In this work, activity of each agent is determined when its fitness is compared against a random agent (which is different from the selecting one); if the selecting agent has a better fitness (smaller value in minimisation problems) than the randomly selected agent, it will be flagged as active, otherwise inactive. Higher rate of inactivity boosts exploration, whereas a lower rate biases the performance towards exploitation.

Algorithm 2 Passive Recruitment Mode

```
01: For each agent ag
02:    If ( !ag.isActive )
03:       r_ag = pick a random agent
04:       If ( r_ag.isActive )
05:          ag.hypothesis = r_ag.hypothesis
06:       Else
07:          ag.hypothesis = generate random hypothesis
08:    End If
09: End For
```

B. Partial Function Evaluation

One of the concerns associated with many optimisation algorithms (e.g. Genetic Algorithm [3], Particle Swarm Optimisation [4] and etc.) is the repetitive evaluation of a computationally expensive fitness functions. In some applications, such as tracking a rapidly moving object, the repetitive function evaluation significantly increases the computational cost of the algorithm. Therefore, in addition to reducing the number of function evaluations, other measures can be used in an attempt to reduce the computations carried out during the evaluation of each possible solution, as part of the overall optimisation (or search) processes.

The commonly used benchmarks for evaluating the performance of swarm intelligence algorithms are typically small in terms of their objective functions computational costs [5], [6], which is often not the case in real-world applications (examples of costly evaluation functions are seismic data interpretation [6], selection of sites for the transmission infrastructure of wireless communication networks and radio wave propagation calculations of one site [7] etc.).

Costly objective function evaluations have been investigated under different conditions [8] and the following two broad approaches have been proposed to reduce the cost of function evaluations:

- The first is to estimate the fitness by taking into account the fitness of the neighbouring elements, the former generations or the fitness of the same element through statistical techniques introduced in [9], [10].
- In the second approach, the costly fitness function is substituted with a cheaper, approximate fitness function.

When agents are about to converge, the original fitness function can be used for evaluation to check the validity of the convergence [8].

The approach that the standard SDS algorithm uses is similar to the second method. Many fitness functions are decomposable to components that can be evaluated separately. During the test phase of SDS, in partial function evaluation (pFE, which is some function of the agent's hypothesis; $pFE = f(h)$), the evaluation of one or more of the components may provide partial information to guide the subsequent optimisation process.

III. Clustering Problem

Clustering is often described as the unsupervised classification of patterns into groups or clusters. Considering the many data analysis techniques, data clustering approach is among the most popular.

Clustering, which is often described as the unsupervised classification of patterns into groups or clusters, is the process through which data are grouped according to their similarity; therefore each entity in one group or cluster is believed to have the most similarity to the members in the same group and largely dissimilar to entities from other groups; measuring distance between data object is an indicator of similarity.

The following poses the problem in a minimisation form, where N number of data objects are to be allocated to M cluster and the goal is to minimise the sum of the squared Euclidean distance between each object and the centre of the related cluster.

$$F(D,C) = \sum_{i=1}^{\mathrm{N}} \sum_{j=1}^{\mathrm{M}} w_{ij} \parallel D_i - C_j \parallel^2 \qquad (1)$$

where $\parallel D_i - C_j \parallel$ return the Euclidean distance between the cluster centre C_j and data object D_i. M and N represent the the number of clusters and data objects respectively. The association weight of data object D_i with the cluster j is given as w_{ij}; this figure is set either in the binary form of 0 and 1 (i.e. $w_{ij} = 1$ if object i is allocated to cluster j, otherwise $w_{ij} = 0$), or in fuzzy clustering, the value of w_{ij} belongs to the interval $[0, 1]$.

In another form, and as reported in [11], clustering is defined as stated below:

Let $O = \{o_1, o_2, ..., o_n\}$ be a set of n objects and let $X_{n\times p}$ be the profile data matrix, with n rows and p columns. Each *i*th objects is characterized by a real-value *p*-dimensional profile vector $\vec{x}_i$, where each element x_{ij} corresponds to the *j*th real-value feature $(j = 1, ..., p)$ of the *i*th object $(i = 1, ..., n)$. Given $X_{n\times p}$, the goal of a clustering algorithm is to determine a partition $G = \{C_1, C_2, ..., C_k\}$ (i.e., $C_g \neq \Phi, \forall g; C_g \cap C_h = \Phi, \forall g \neq h; \cup_{g=1}^{k} C_g = O$) such that objects belonging to the same cluster are as similar to each other as possible, while objects which belong to different clusters are as dissimilar as possible. It has been shown that the clustering problem is NP-hard when the number of clusters exceeds three [12].

A. Iris dataset

In this work *Iris* dataset from Machine Learning Laboratory [13] is used. Arguably, the Iris dataset is one of the most well-known and frequently used datasets in pattern recognition and clustering literature; one of the most cited works in this context is a paper written by Fisher in 1936 [14] .

In this dataset, there are three iris types each of which have 50 samples; each type represents one group of iris plant where one out of three types is linearly separable from the other two. Each of the 150 samples in the dataset has four numeric features representing the sepal length, sepal width, petal length and petal width (all in cm). There are no missing attributes in any of the samples. See Table I for an overview of the content of the iris dataset which is visualsed in Fig. 1.

B. Other techniques used for comparison

In order to compare the performance of SDS against other methods, nine other techniques are briefly explained below which be used in the next section:

Support Vector Machine (SVM) : SVM is one of the most well-known and successful classifiers. SVM [15] is a kernel method based on a support vector description of a data set consisting of positive examples only. If all data are in-liers, one-class SVM computes the smallest sphere in feature space enclosing the image of the input data.

Neural Gas: The neural gas – an artificial neural network, introduced in 1991 – is a simple algorithm for finding optimal data representations based on feature vectors. The algorithm was coined "neural gas" because of the dynamics of the feature vectors during the adaptation process, which distribute themselves like a gas within the data space [16].

K-means : Among the classical clustering algorithms, K-means [17] is the most well known algorithm due to its simplicity and efficiency. In K-means the centres are moved repeatedly by computing, at each iteration and for each center, the smallest sphere enclosing the closest data until no center changes anymore.

Ng-Jordan : Ng Jordan, a spectral clustering algorithm, is built upon the earlier work of Weiss and Meila and Shi, who analysed algorithms that use k eigenvectors simultaneously to perform the tasks [18].

Self-Organising Map (SOM) : A self-organising map [19] or self-organising feature map (SOFM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretised representation of the input space of the training samples, called a map. This makes SOMs useful for visualising low-dimensional views of high-dimensional data.

Particle Swarm Optimization (PSO): This algorithm was developed based on the swarm behaviour, such as fish schooling and bird flocking [20]. In PSO, population are evolved by moving the candidate solutions around in the search space using the best found locations, which are updated as better locations are found by the candidates.

Gravitational Search Algorithm (GSA) : In the GSA [21] algorithm, the searcher agents are a collection of masses that interact with each other based on the Newtonian gravity and the laws of motion [22].

Big Bang-Big Crunch (BB-BC) : BB-BC optimisation [23] is based on one of the theories of the evolution of the universe. It is composed of the big bang and big crunch phases. In the big bang phase the candidate solutions are spread at random in the search space and in the big crunch phase a contraction procedure calculates a center of mass for the population.

Black Hole algorithm (BH): Similar to other population-based algorithms, BH [24] starts with an initial population of candidate solutions to an optimisation problem and an objective function that is calculated for them. At each iteration of the black hole algorithm, the best candidate is selected to be the black hole, which then starts pulling other candidates around it, called stars. If a star gets too close to the black hole, it will be swallowed by the black hole and is gone forever. In such a case, a new star (candidate solution) is randomly generated and placed in the search space and starts a new search.

In the next section, the clustering performance of SDS algorithm over Iris dataset is compared against above-mentioned nine techniques.

IV. Experiments

Here the process through which SDS algorithm is adapted to perform the clustering tasks is detailed and the steps taken during the *test* and *diffusion* phases are explained. In order to apply this swarm intelligence algorithm to the dataset the following are considered:

- **Search space** is the entire Iris dataset
- **SDS hypothesis** refers to an iris element
- **Iris attributes:** Each iris has four attributes (i.e. sepal length, sepal width, petal length and petal width; see Table I and Fig. 1).
- **Micro-features:** The four attributes of each Iris are considered the micro-features of the hypothesis. Therefore each SDS hypothesis has four micro-features which refer to the attributes of the iris.

As stated below, in order to visualise the clustering process, the irises' four attributes are illustrated in Fig. 1, where the first five-rows belong to the first cluster of Iris dataset, and the second and third five-rows belong to the second and third clusters of the Iris dataset respectively.

A. Applying SDS algorithm

This section details the process through which SDS algorithm conducts the clustering process:

1) Initialisation phase: During the initialisation phase, one iris is chosen randomly from the dataset and is set as a model. Then each agent is randomly associated with an iris from the search space.

2) Test phase: During the test phase, each agent (which is already allocated to an iris) randomly picks one of the four micro-features and compares its value against that of the model. If the difference between the two corresponding micro-features is within a specific threshold, τ_d (where τ is the threshold and d is the dimension) the agent becomes active, otherwise inactive.

TABLE I. IRIS DATASET

	Iris setosa				Iris versicolor				Iris virginica			
	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width
1	5.1	3.5	1.4	0.2	7	3.2	4.7	1.4	6.3	3.3	6	2.5
2	4.9	3	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
3	4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3	5.9	2.1
4	4.6	3.1	1.5	0.2	5.5	2.3	4	1.3	6.3	2.9	5.6	1.8
5	5	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3	5.8	2.2
6	5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3	6.6	2.1
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
49	5.3	3.7	1.5	0.2	5.1	2.5	3	1.1	6.2	3.4	5.4	2.3
50	5	3.3	1.4	0.2	5.7	2.8	4.1	1.3	5.9	3	5.1	1.8

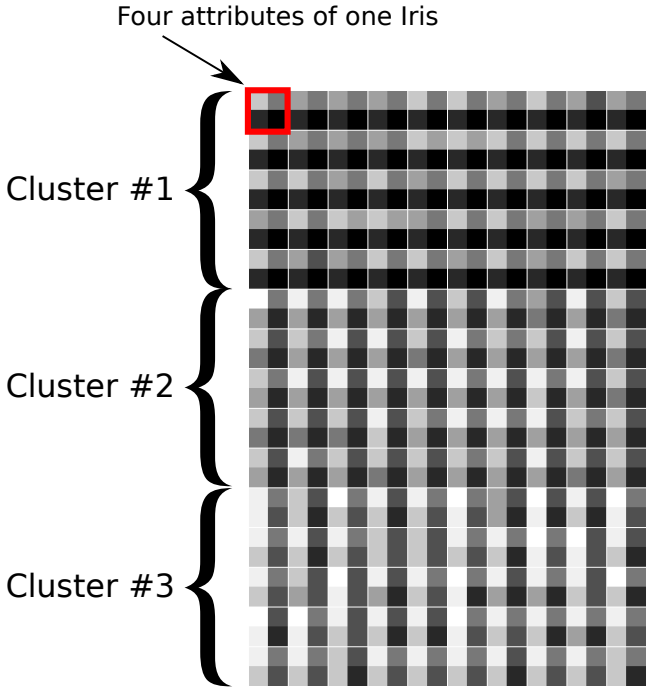

Fig. 1. Iris dataset has three clusters; and each iris elements has four attributes.

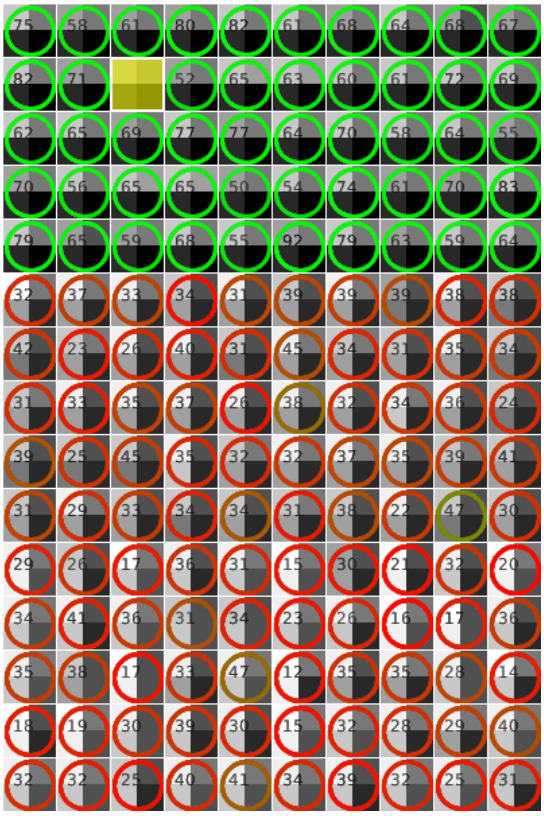

Fig. 2. Agents are initialised throughout the search space; the red circles signify the inactive agents and the green circles highlight the active ones.

3) Diffusion phase: The process in the diffusion phase is the same as the one detailed in the algorithm description: each inactive agent picks an agent randomly from the population; if the randomly selected agent is active, the inactive agent adopts the hypothesis of the active agent (i.e. they refer to the same iris as their hypothesis), otherwise the inactive agent picks a random iris from the dataset.

4) Categories, Clusters and Termination: The agents iterate through the test and diffusion phases again until all agents are active. At this stage, the irises referred to by all the active agents are assigned to a category. Additionally, the number of active agents on each iris is logged.

Once a category is determined, the process is repeated from the initialisation phase where agents are initialised throughout the search space and the first iris which has not yet been assigned to any categories is set as the new model. Then the algorithm iterates through the test and diffusion phases until all irises are allocated to a category.

Finally, categories form the clusters, and when there exist irises that belong to more than one cluster, they will be allocated to the one which has attracted a larger number of active agents.

B. Results and Discussion

This section details the clustering performance of SDS and provides the summary of a comparison conducted amongst SDS and five other clustering techniques.

As mentioned earlier, after the initialisation phase, the algorithm iterates between the test and diffusion phases in order to determine the categories. The only tunable parameters for SDS is the swarm size, N which is empirically set to $N = 10,000$. Threshold, $\vec{\tau}$, which is the acceptable distance between the model and other samples for each dimension, d, is calculated using the following formula:

$$\vec{\tau}_d = \sum_{t=1}^{c} \left| \frac{\text{MAX}\left(\vec{I}_{id}^{t}\right) - \text{MIN}\left(\vec{I}_{id}^{t}\right)}{c} \right| \qquad d = 1, 2, 3, 4 \quad (2)$$

where c is the number of iris types in the dataset (or the number of clusters); $\vec{I}_{id}^{t}$ represents the value of i^{th} iris with type t and dimension d. There are 3 iris types and the dimensionality of the problem is 4 (i.e. Sepal and Petal's lengths and widths). Therefore the difference between the minimum and maximum values in each iris type is calculated, then the sum of the differences in each dimension is averaged and used to calculate the threshold. Using the formula above the threshold is set to $\vec{\tau} = \{2.2, 1.7, 1.8, 0.8\}$. Note that not the entire data in each iris type needs to be used in the formula.

As illustrated in Fig. 2, active agents are shown in green and inactive agents are presented in red; the randomly selected model is highlighted in yellow[2]. Given that the model is on the first cluster, it is likely that the active agents have a bigger presence in this cluster. The numbers in each block signify how many active agents there are.

As can be observed in the figure, in some of the blocks while there might exist a high presence of active agents (e.g. row six and column one, which has attracted 32 active agents) there is also a high number of inactive agents resulting in the colour not being bright green. The reason why an iris could make an agent active and another one inactive can be explained through SDS's micro-features: each block consists of four micro-features (the same as the number of attributes in each Iris sample) and there are cases when, for example, two out of four micro-features are within the acceptable threshold from the model while the other two are outside the range; therefore if an agent picks one of the micro-features that are within the threshold, it becomes active, but if it randomly picks one of the other micro-features, it becomes inactive. Duducing from this, it is evident that having more micro-features within the range of the model results in more agents becoming (or rather staying) active, and as a result forming a stable category.

In one trial, the formation of the three categories is shown in Fig. 3. And the final clusters are displayed in Fig. 4.

The Iris samples in the upper third part of the search space (see Fig. 1) belong to *Iris setosa* type which is linearly separable from the other two iris types (i.e. *Iris virginica* and *Iris versicolor*). This separability is clearly highlighted in Fig. 3-top where the model is in the first cluster and the entire agents population are attracted to this cluster. As illustrated in the middle and bottom figures of Figs. 3, the two other two iris types are not linearly separable.

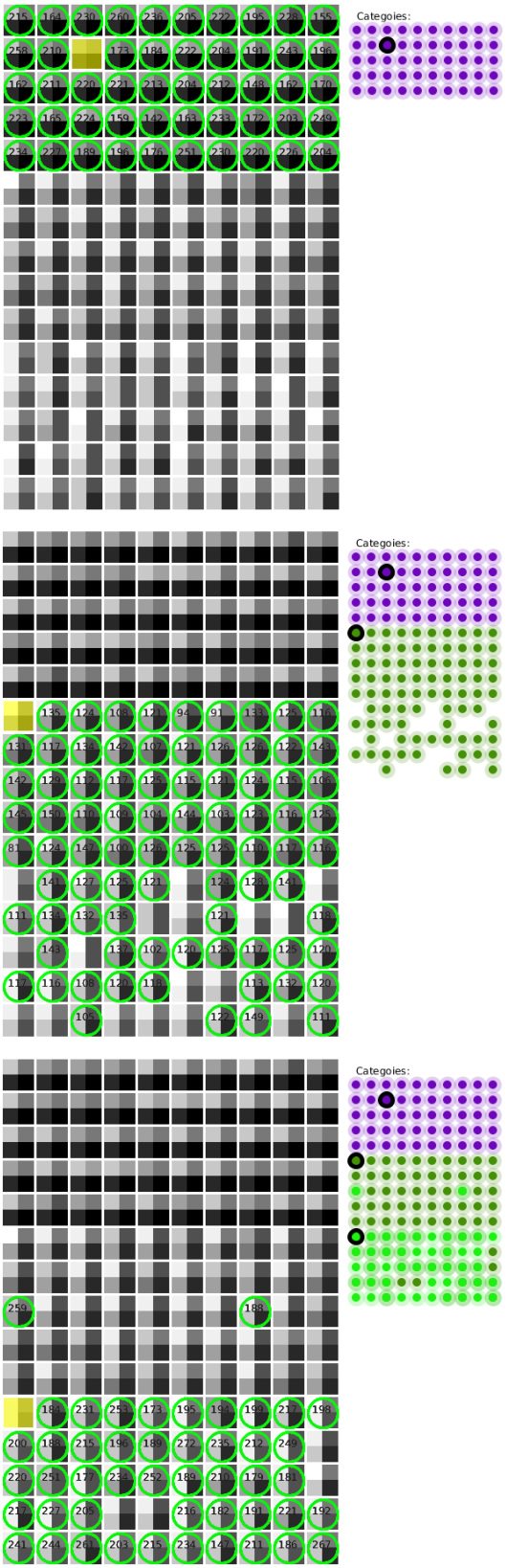

Fig. 3. The process through which the first, second and third categories are formed.

[2]Note that there is an alpha value for the transparency of the colours. Therefore there could be an overlap between the green and red colours in some of the blocks.

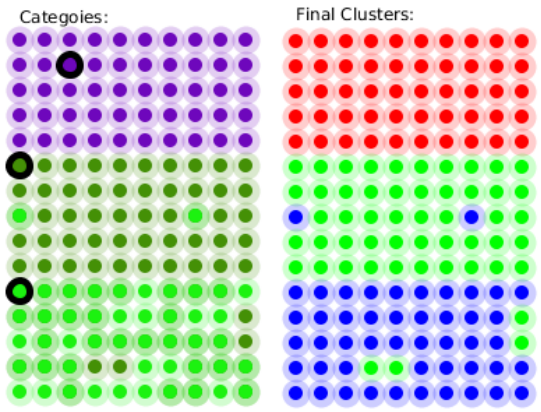

Fig. 4. Final clusters identified.

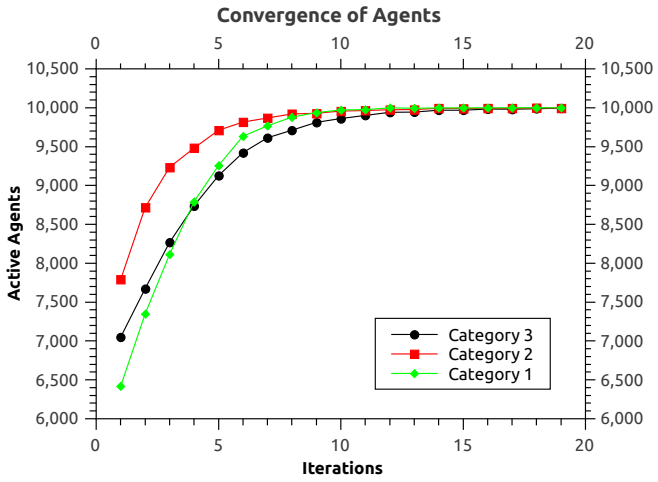

Fig. 5. Convergence of Agents.

In this section, the result of running SDS on the Iris dataset is reported. This experiment consists of 50 trials. The average number of iterations needed to cluster the Iris dataset is 25 ± 4. Table II summarise the error in SDS clustering activity. The table shows that SDS is always successful to classify the first cluster with 0% error; the misclassification in clusters B and C are 10% and 6% respectively.

TABLE II. CLUSTERING ERROR

	Misclassification	Percentage
Cluster A	0 ± 0	0%
Cluster B	5 ± 3	10%
Cluster C	3 ± 2	6%
Total	8	5.33%

Table III contains information about the number of Irises in each categories as well as the number of iterations needed for each category to be finalised (reaching the max agents activity level). Also the standard deviation of the agent's activity for each Iris is reported. This table demonstrate that cluster 1 which has 50 Irises in each trial is easily separable from the others, while in categories 2 and 3, the activity of agents should be taken into account before determining the clusters.

TABLE III. AGENTS ACTIVITY IN EACH CATEGORY

	Cat #1	Cat #2	Cat #3
Irises in Categories	50 ± 0	83 ± 3	48 ± 4
Number of Iterations	19 ± 4	25 ± 2	30 ± 6
Std Dev of activity	96.13	61.02	99.13

Fig. 5 illustrate the behaviour of the agents as they converge to either of the three categories. This figure shows that category 1, at the initial point, has the least number of active agents while category 2 has the highest number of initial active agents due to its similarity with several irises in the third cluster in addition to its own cluster (the second cluster). See Fig. 3-Middle which shows the number of Irises attracted to the second category.

Additionally, Figs. 6 are presented in order to provide further details about the activity of agents during the formation of each categories. For example, Fig. 6-Top, in addition to showing the total number of active agents at each iterations (while creating the first category), the number of active agents in each cluster is also illustrated; it is is shown that the number of active agents in cluster 1 increases immediately after the start of the clustering process; however, on the contrary to the behaviour of agents in cluster 1, the agents in clusters 2 and 3 lose their activity rapidly[3].

TABLE IV. COMPARISON WITH OTHER DATA CLUSTERING TECHNIQUES

	Correctly Classified	Accuracy Ratio	Ranking
BB-BC	135	89.95 %	6
BH	135	89.98 %	4
GSA	135	89.96 %	5
K-Means	134	89.33 %	8
Neural Gas	138	91.70 %	3
Ng Jordan	126	84.30 %	9
PSO	135	89.94 %	7
SOM	122	81.33 %	10
SDS	142	**94.67 %**	1
SVM	139	92.67 %	2

Following the explanation of the SDS behaviour in clustering the Iris dataset, next, a comparison is presented where the proposed technique is contrasted against nine other clustering technique. In this comparison, the number of correctly classified Irises is detailed as well as the accuracy ratio and the ranking of the techniques. See Table IV for the comparison. Some of these techniques are designed solely for clustering purposes and some are optimisation techniques adapted to address clustering problems[4].

These results show the promising performance of SDS compared to the aforementioned nine classifiers. In this dataset, SDS outperforms all the classifiers including SVM (among the most well-known and successful classifiers). One of the

[3]Here is the link to the video of the clustering process at a reduced speed: http://youtu.be/v81b2_sUbys

[4]The figures (except SDS's) are borrowed from [25], [24], [26].

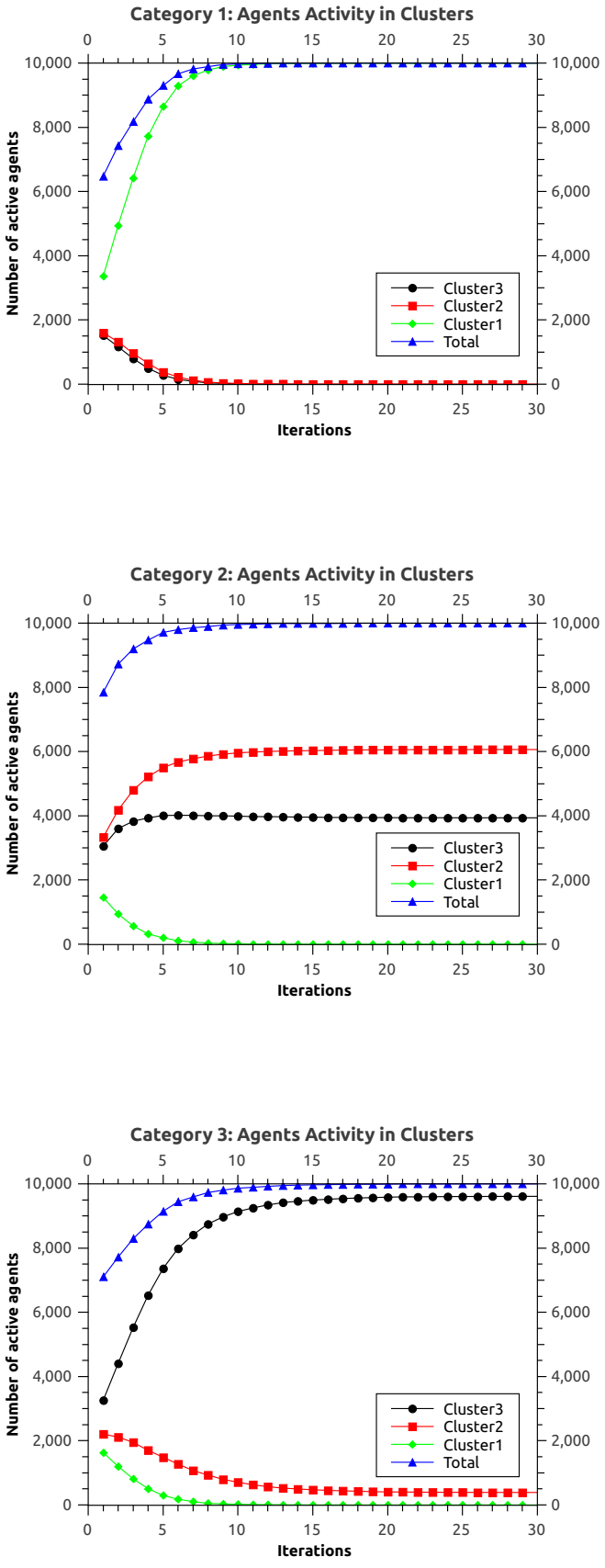

Fig. 6. Convergence of Agents.

main features of SDS is the embedded partial function evaluation which leads to inexpensive and low cost comparison of samples. In SDS's partial function evaluation, at any time, each agent only checks only one of the features of the sample, thus the comparison is computationally inexpensive. In other word, in the notion of partial-function evaluation, a given set of parameter values (the agent hypothesis) a complex objective function is broken into m components, only one randomly selected of which will be evaluated and the subsequent agent-activity is based on this. Clearly, as this process merely evaluates $1/m$ of the total number of computations required for the full hypothesis evaluation, it concomitantly offers a potentially significant performance increase. This feature is particularly important when dealing with problems with very large dimensionality including fMRI related classifications which are the subject of ongoing research.

V. Conclusion

This paper deployed Stochastic Diffusion Search algorithm (SDS) in a novel approach for clustering purposes. This algorithm has been successfully applied to several optimisation problems as well as medical imaging. In this work, the performance of SDS is contrasted against several other classic classifiers including Support Vector Machine (SVM) and K-Means. SDS is shown to outperform the nine techniques referred to in this paper in clustering the Iris dataset. The only tunable parameter for SDS is the swarm size; the values of the threshold vector, $\vec{\tau}$ are generated using a formula suggested in the paper. Given the partial function evaluation feature of SDS and the low computational cost of comparing samples, this algorithm is likely to be particularly useful when applied to problems with huge dimensionality. Topics for future research are dynamic fine tuning of the threshold as well as the application of the introduced technique to real-world and high-dimensional fMRI problems.

References

[1] J. Bishop, "Stochastic searching networks," in *Proc. 1st IEE Conf. on Artificial Neural Networks*, London, UK, 1989, pp. 329–331.

[2] M. M. al-Rifaie and M. Bishop, "Stochastic diffusion search review," in *Paladyn, Journal of Behavioral Robotics*. Paladyn, Journal of Behavioral Robotics, 2013, vol. 4(3), pp. 155–173.

[3] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[4] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. IV. Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948.

[5] J. Digalakis and K. Margaritis, "An experimental study of benchmarking functions for evolutionary algorithms," *International Journal*, vol. 79, pp. 403–416, 2002.

[6] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias, "Evaluating evolutionary algorithms," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 245–276, 1996.

[7] R. Whitaker and S. Hurley, "An agent based approach to site selection for wireless networks," in *1st IEE Conf. on Artificial Neural Networks*. Madrid Spain: ACM Press Proc ACM Symposium on Applied Computing, 2002.

[8] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *In: Soft Computing*, vol. 9, pp. 3–12, 2005.

[9] J. Branke, C. Schmidt, and H. Schmeck, "Efficient fitness estimation in noisy environments," *In Spector, L., ed.: Genetic and Evolutionary Computation Conference, Morgan Kaufmann*, 2001.

[10] M. A. el Beltagy and A. J. Keane, "Evolutionary optimization for computationally expensive problems using gaussian processes," in *Proc. Int. Conf. on Artificial Intelligence'01*. CSREA Press, 2001, pp. 708–714.

[11] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4761–4767, 2010.

[12] P. Brucker, "On the complexity of clustering problems," in *Optimization and operations research*. Springer, 1978, pp. 45–54.

[13] C. Blake and C. J. Merz, "{UCI} repository of machine learning databases," 1998, available from: http://www.ics.uci.edu/-mlearn/MLRepository.html. [Online]. Available: http://www.ics.uci.edu/-mlearn/MLRepository.html

[14] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[15] D. M. Tax and R. P. Duin, "Support vector domain description," *Pattern recognition letters*, vol. 20, no. 11, pp. 1191–1199, 1999.

[16] T. Martinetz, K. Schulten *et al.*, *A "neural-gas" network learns topologies*. University of Illinois at Urbana-Champaign, 1991.

[17] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[18] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[19] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[20] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 43. New York, NY, USA: IEEE, 1995.

[21] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[22] ——, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[23] O. K. Erol and I. Eksin, "A new optimization method: big bang–big crunch," *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.

[24] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013.

[25] F. Camastra and A. Verri, "A novel kernel method for clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 801–805, 2005.

[26] W. QU, W. WANG, and F. LIU, "A novel classifier with support vector machine based on ap clustering," *Journal of Computational Information Systems*, vol. 9, no. 10, pp. 4041–4048, 2013.